Instructor: Nicolas Savva

March 4th, 2015

\(^1\) based on slides by Hussam Abu-Libdeh, Bruno Abrahao and David Slater over the years
Announcements

- The Final Project is out (due 03/10 at 11:59PM (+2 days))
- Office hours continue until 03/15
- This is our last lecture!
Recap of useful tools and concepts
many things left out, check previous slides
Finding help

Finding help on anything
\texttt{man <command name>}

- You can search in \texttt{man} by pressing the / key and then the keyword you’re searching for
  - find next match by pressing the n key
  - find previous match by pressing N
  - stop search by pressing the Esc key
- Exit by pressing q
Moving around

Listing directory content
ls

Listing everything that begins with 'foo'
ls foo*

Listing everything that ends with .txt
ls *txt

Listing everything inside a subdirectory
ls subdirname/*

Changing directories
cd dirname
File system manipulation

Make new directory
mkdir newdirname

Copy file1 to file2
cp file1 file2

Moving a file to new directory
mv file1 newdir

Change file permissions
chmod u+x myfile

Change file ownership
chown 'newuser:newgroup' myfile
Displaying content

Printing something to output stream (default: screen)
- echo "Hello World!"

Print file content
- cat myfile

Paging file content
- more myfile

Paging with better scrolling
- less myfile

Concatenate multiple files and print them
- cat file1 file2 file3
Input/Output streams

Programs can receive input from an input stream (stream 0 a.k.a STDIN) and produce normal output to an output stream (stream 1 a.k.a STDOUT) and error output to an error stream (stream 2 a.k.a STDERR).

- By default STDIN is just keyboard input from user
- By default STDOUT is just printing to screen

**Important point #1**

We can do many powerful things in Unix by chaining input/output of different commands. So output of one is fed as input to other. This is done via piping (the |)

**Important point #2**

We can redirect these streams to other locations such as take input from a file, or write output to a file. This is done with redirection operators (the < and >)
Redirecting input to be read from a file

```bash
program < file
```

Redirecting output to be written to a file

```bash
program > file
```

Redirecting output to append to a file

```bash
program >> file
```

Redirecting input from file1 and output to file2

```bash
program < file1 > file2
```

More on redirection (such as combining streams) in previous lectures.
Chaining programs using pipes

$$\text{program1} \mid \text{program2} \mid \text{program3}$$

This pipes (i.e. connects) the output of program1 as input to program2, and output of program2 as input to program3.
Remember!

Once an input or output stream is redirected or piped, it is consumed, and you cannot reuse it. So, `program1 > file | program2` does not redirect the output stream from `program2` twice.

Use the `tee` command if you want to do that.
Running programs sequentially

Run program1 followed by program2
program1 ; program2

Run program1 followed by program2 only if program1 terminated successfully
program1 && program2
A bunch of nice tools
Translating

tr SET1 SET2

Does a character by character substitution in the input stream and writes it to output. So $i^{th}$ character in SET1 gets substituted with $i^{th}$ character in SET2.

For more options such as deleting and complementing

man tr

Remember, tr only works with input stream, so to read a file you have to use redirection or piping:

tr [A-Z] [a-z] < myfile

cat myfile | tr [A-Z] [a-z]
A pattern is a list of characters that satisfy some conditions.

**Example**

The pattern "Shark" matches anything that contains an S followed by an h then an a then an r then a k.

We get more flexibility by using options, wild cards, and repetition.

- `ca[rt]` matches car and cat but not cart
- `car*` matches ca and car and carr and carrr ..
- `car[0-9]` matches car0 and car1 and car2 .. and car9
grep PATTERN FILE

grep looks for lines in FILE that match PATTERN and print the whole line.
Many many more options

man grep

The manual page for grep also contains a good section about using regular expressions for patterns.
## Some grep flags

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print only matching segments</td>
<td>grep -o</td>
</tr>
<tr>
<td>Print only non-matching lines</td>
<td>grep -v</td>
</tr>
<tr>
<td>Ignore case</td>
<td>grep -i</td>
</tr>
<tr>
<td>Get only full word matches</td>
<td>grep -w</td>
</tr>
<tr>
<td>Get pattern list from a file</td>
<td>grep -f patternsFile</td>
</tr>
<tr>
<td>Print names of files that contain matches</td>
<td>grep -l</td>
</tr>
</tbody>
</table>
sed is a stream editor. You can simply use it to do substitutions in streams of data based on pattern matching.

**Simple usage**

```
sed 's/pattern to match/what to substitute in/' myfile
```

**Example: substitute hot dog with hamburger**

```
sed 's/hot dog/hamburger/g' menu.txt
```

**Reverse phone book name order**

```
sed -r 's/([A-Z]+), ([A-Z]+)/\2 \1/'
```

**For more info**

```
man sed
```
More powerful stream manipulation

gawk allows you to read lines, break them into fields, match patterns, and do arithmetic based on that.

Example

gawk '
    /apple/ {count += 1; print "found an apple"}
    END {print "Total apples=", count}
' shopping_log.txt

Goes line by line, if line contains a match for apple, a count is incremented, and a message is printed. At the end, the total apple count is printed.

Remember, this checks every line. No needs for loops!

/pattern/{command}

As always

man gawk
Things we covered in this course

- The Unix Filesystem (cd, cp, mv, rm, ls, chmod)
- Managing Process and Jobs (ps, top, fg, bg, kill)
- IO: Pipes and Streams (|, <, >, >>)
- Remote Resources (scp, sftp, ssh, wget, curl, lynx)
- vim and nano
- tmux and screen
- Regular Expressions
- Shell Scripting (variables, loops, test expressions, arg parsing)
- Using arrays and functions
- grep, sed and awk
- Intro to Python (and Perl)
- Automation, make and VCS (cron, crontab, git, svn)
- many other tools...
### Where do we go from here?

#### Programming Languages
- CS 2022: Introduction to C
- CS 2024: C++ Programming
- CS 3110: Data Structures and Functional Programming

#### Systems
- CS 3410: Computer System Organization and Programming
- CS 4410: Operating Systems

#### down the line...
- CS 5152: Open-Source Software Engineering
- CS 5220: Applications of Parallel Computers
- CS 5412: Cloud Computing
Useful websites

- http://www.gnu.org/
- http://www.linuxfoundation.org/
- http://confluence.atlassian.com/display/BITBUCKET/Bitbucket+Documentation+Home
- http://linux.com
It has been a fun short course.
I hope you found it useful and that you learned some cool new things.

Important takeaway

“man” is your best friend :-)

That’s all folks!
Thank You